```
/* 03/05/03 *this is source code in C programming language of
"pinned" program running on remote machine~~/~~
/* by Slava Barsuk */
/* on demand power reset          */

#include <stdio.h> definition of miscellaneous C headers
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/select.h>
#include <sys/reboot.h>
#include <sys/sched.h>
#include <sys/lock.h>
#include <netinet/in.h>
#include <netdb.h>
#include <spc.h>
#include <strings.h>
#include <string.h>
#include <signal.h>


char cws_name[32]; defninition of data structures
struct    sockaddr_in    server;
int  sock,ws;

int  main_processing() body of subroutine to perfom power
operation, called from main body, when request comes on tcp
socket
{
static    struct    sockaddr_in    *pfrom; defninition of data
structures
static    struct    sockaddr  from;
static    struct    hostent        *hp;
static    struct
{ deninition of memory buffer for received request, consists of 3
elements - len, code and text
int  len;
int  code;
char text[24];

} buf;

static    int  addrlen,NB;

     addrlen=sizeof(from);
     pfrom=(struct sockaddr_in *)&from;
     NB=read(ws,&buf,sizeof(buf)); read request from tcp socket
ws into memory reffered as buf. NB receives number of actual
bytes read
```

```
     if(NB!=8 || buf.len!=4 ) return(-1); Check that number of
bytes read is 8 (NB==8) and len element is equal 4. If not,
return to main body and continute listening ( ignore request)

     if(getpeername(ws,&from,&addrlen)>=0) get tcp address of
request sender
       {
          hp=gethostbyaddr(&pfrom->sin_addr,4,AF_INET); resolve
tpc address of request sender into symbolic hostname
          if(hp==NULL) return(-1); return to main body, if unable
to resolve name
          if(strcmp(hp->h_name,cws_name)!=0) return(-1); compare
requester name with authorised hostname, if not, return to main
body (ignore request)
          if( buf.code==12 ) check message code. if 12, initiate
reboot operation
            {
               reboot(RB_SOFTIPL); system call to reboot
             }
            else if( buf.code==13 ) if message code is 13,
initiale power off (halt) operation
            {
               reboot(RB_HALT); system call to halt
            }
       }
}

void main(int argc,char *argv[]) main body
{

struct    servent *port,*getservbyname(); defninition of data
stuctures
int     l;

actual code starts here
     strncpy(cws_name,argv[1],30); accept authorized hostname as
parameter
     if(strlen(cws_name)<2) exit(6); check that authorized
hostname is not empty, exit program if name is not provided
     port=getservbyname("pwrport",0); if(port==0) exit(4);
resolve tpc communication port, exit program if  port can't be
resolved

     sock=socket(AF_INET, SOCK_STREAM,0); create and initialize
tcp socket structure for communication
     if (sock<0)  exit(5); exit program if socket can't be
created

     server.sin_family=AF_INET;
     server.sin_len=sizeof(server);
     server.sin_addr.s_addr=INADDR_ANY; set listener address
(any)
     server.sin_port=htons(port->s_port); set listener port
     l=sizeof(server);
     if (bind(sock,(struct sockaddr *)&server, l)) bind socket to
tcp port, exit if can't bind
                exit(7);

     if (getsockname(sock, (struct sockaddr *)&server, &l))
          exit(7); check that socket was created and binded
succesfuly
     plock(TXTLOCK); pin program to memory ( claim 1)

     listen(sock,10); start listening to requests on tcp socket
sock ( claim 1)

     do { start loop to wait and process requests (claim 1)
          ws=accept(sock,0,0); wait for request to come and
create communication socket ws for it, when it came (claim 1)
          main_processing(); peform request analysys and
processing ( subroutine main processing, which does power
operation)
          close(ws); close socket
        }
     while(1); go to the beginning of the loop ( keep waiting for
new requests to come)

}
```